An Attempt at Developing a Crown Fire Ignition Model

FINAL REPORT

An Attempt at Developing a Crown Fire Ignition Model

by

Steve Izbicki
Heat Transfer Engineer

and

Robert Keane
Quantitative Ecologist

Submitted Jim Brown, INT Research Station for partial completion of Cooperative Agreement INT-88352-COA

November 17, 1989

INTRODUCTION

Crown fires are perhaps the most dangerous and most severe form of forest fire behavior possible in forests and shrublands around the world (Rothermal and Mutch 1986). Although these crown fires (fires which propagate through tree and shrub crowns) typically represent less than 10% of the approximately 300,000 fires suppressed in the United States and Canada each year, they are easily the most damaging (Albini 1984). Crown fires spread very quickly and produce tremendous amounts of heat which often cause the loss of human life (Beighley and Bishop 1988, Rothermal and Mutch 1986). Crown fires usually burn large areas of land and this burned land can be extensively damaged (Albini and Stocks 1986). Modelling and predicting the start and spread crown fires would greatly aid in the saving of human life and forested lands.

Many crown fire prediction models have been used with varing degrees of success (Van Wagner 1977, Albini 1988, Albini and Stocks 1986). However, previous models have a limited applicability because most 1) resort to empirically based solutions, 2) model only flow field to crown not heat transfer, or 3) predict crown spread from ground fuel parameters.

The purpose of this study was to develop a crown fire ignition model that would predict when a ground fire would ignite a tree crown. The approach used for the modelling effort was to simulate crown fire ignition based on general fluid flow theory, heat transfer principles, and numerical solution techniques. This model was designed so it could be inserted into the fire succession model FIRESUM (Keane and others 1989) to predict effect of successive crown fires on forested ecosystems.

Study Objective

Our modelling goal was to develop a broad based crown fire ignition model applicable to any forest or shrub environment but limited to the confines of

the succession model FIRESUM. The method used was to calculate convective and radiative heat transfer from an advancing flame front to the crown fuel. The crown was assumed to ignite when crown temperature was raised above some ignition threshold value.

Model Background

The spread of fire by radiation through fuel beds on the ground has been thoroughly investigated by Albini (1985) and Telisin (1974). Albini (1986) modeled fire spread through fuel beds where unignited fuel was heated by radiation and cooled by reradiation and convection. It follows that crown fire potential can then be assessed once the heat source is quantified on the ground (i.e. surface fire).

Mathematical modelling of thermally driven flow fields in viscous fluids is well documented (Bodoia and Osterle 1962, Williams 1967, Quon 1972). Work by Luti and Brzustowski (1977) has shown the possiblity of predicting the flow field due to a strong heat source in the atmosphere with a uniform cross wind. Additional work by Luti (1980) simulates temperature and flow field development in the atmosphere above a uniform high temperature source and cross wind.

METHODS

Model Development

The crown fire model uses non-dimensional, finite differenced-central difference, incompressible Navier-Stokes and energy equations to calculate a temperature and velocity (wind) field above an advancing fire front in a uniform cross wind. Once the temperature and velocity fields are computed, the model calculates radiative and convective heat transfer to tree crowns using the Stefan-Boltzmann equations and Dittus-Boelter equations, respectively (Incropera and Dewitt 1981). The crown ignites when crown temperature is

raised above a threshold ignition value. The following steps were used to compute crown ignition (see figs 1a and 1b for flow charts):

1. A two dimensional grid was placed above a simulated fire with the bottom gridline even with the ground surface (fig. 2). Distance between the gridpoints (dx on the horizontal, dy on the vertical) is variable (user-input as 1 meter for this study).

2. A steady state fire-induced temperature and velocity field was calculated for the grid points (fig. 1a). The forward time/centered space (FTCS) method (Roache 1972) was used to develop the equations that compute temp-velocity field (fig. 3). These equations were solved simultaneously using iterative techniques (Roache 1972). A fourth order smoothing function was employed to facilitate convergence and minimize instabilities in the finite differenced equations (Tannahill and others 1984). It was assumed the flame was a STATIONARY uniform heat source for equation solution. Boundary conditions for equation solution is shown in fig. 4.

3. Radiative and convective heat transfer to two points on the tree crown was calculated for each horizontal grid point (fig. 5). To simulate the advancing flame front, trees were moved through the temp-velocity field at the input flame spread rate (see fig. 1b). The two points on the tree crown idicate the start of the live crown and the start of the dead crown (fig. 6).

4. Crown ignition was determined by calculating heat needed for ignition and comparing it to the heat transfered by the fire. The heat needed for ignition was calculated by summing 1) heat required to raise crown fuel moisture to boiling point, 2) heat needed to evaporate crown fuel moisture, and 3) heat needed to raise dry crown fuel to ignition point (fig. 7). The tree crown ignited if the amount of heat transferred to crown exceeded the heat required for ignition.

This last two steps were repeated for each tree in FIRESUM's simulation plot (Keane and others 1989). A tree was considered dead if its crown ignited. These procedures were implemented in a FORTRAN program call CROWN and tested with microcomputers (Appendix A).

RESULTS AND DISCUSSION

Computer simulation results from the crown fire ignition model were encouraging but not as successful as hoped. The FTCS finite difference calculation of the temp-velocity grid did not consistently converge. Through the first 100-200 iterations of the solution, the temp-velocity grid developed

as expected. Then, for reasons as yet unexplained, the solution diverged and flow field values approached infinity.

The divergence could be due, in part, to the solution assumptions. The magnitudes of temperature and velocity change around the fire might be too large for the grid size. The grid was assumed large enough so that flame effects could be computed within grid boundaries. However, boundary conditions were chosen such that the velocity vector was a constant both at the downstream and upstream boundaries. Therefore even though the gridsize was large (1 meter), the boundary conditions constrained development of temperatures and velocities.

The solution technique could also be the cause of the lack of convergence. Instabilities are common in finite difference equations and these equations are often adjusted so instabilities do not propagate. Rather than using forward-time / centered space solution scheme, an upstream weighted scheme might prove better suited for this particular set of difference equations (Roache 1972). The smoothing function used to minimize equation instability might not have been appropriate. Lastly, constants used in the solution process may need adjustment to improve computer efficiency and guarantee successful convergence.

Several procedures were used to insure algorithm convergence, and unfortunately, all these techniques failed. First, the time step in the solution equations was varied according to equations in Roache (1972, page 52). This resulted in either solution divergence (as before), or extremely long computer run times (38 hours for one execution). Second, the temp-velocity grid around the flame was adjusted so temperature differences were not so severe across grid points. This did not seem to help. Then an unsuccessful attempt was made at adjusting the grid size around the flame so

more points could be represented in grid areas where temperature differences were large. Lastly, the grid size was modified to equal the squareroot of the sum of the squares of the flame length and flame zone. This again seemed to fail.

Since the grid solution seldom converged, heat transfer to the crown could not be estimated. Some preliminary work using a makeshift grid developed by the authors indicate this portion of the program performs as expected. Once the convergence problem is solved the program can be included in the process model FIRESUM.

CONCLUSIONS

Results of this study indicate the concept attempted is viable and further investigation is strongly recommended. A stability analysis should be performed to determined whether the form of equations being used can converge for this problem. The grid size, smoothing function, solution constants, and solution techniques can be modified based on the results of the stability analysis. Special attention should be given to maximizing computer efficiency to achieve successful equation convergence. Suggestions from other scientists include offsetting the vorticity grid to half the grid size from the energy grid, 2) use a different smoothing function, 3) make the grid large (100 points by 100 points) and the grid size small (0.1 m) which means improving the means of computing the solution (larger computer or better programming), 4) make the grid size variable depending on location within the grid, and 5) use a new solution technique. All of these suggestions should be investigated.

REFERENCES

Albini, F.A. 1984. Wildland fires. American Scientist 72: 590-599.

Albini, F.A. 1985. A model for fire spread in wildland fuels by radiation. Combustion Science and Tech. 23:165-175.

Albini, F.A. 1986. Wildland fire spread by radiation - a model including fuel cooling by natural convection. Combustion Science and Tech. 45:101-113.

Albini, F.A. and B.J. Stocks. 1986. Predicted and observed rates of spread of crown fires in immature jack pine. Combust. Sci. and Tech. 48:65-76.

Anderson, D.A. and J.C. Tannahill. 1984. Computational fluid mechanics and heat transfer. Hemisphere Publishing, New York, New York. 115 p.

Beighley, M. and J. Bishop. 1988. Fire behavior on the Fayette fire, or "outside the model". Report on file at Intermountain Fire Sciences Lab, Missoula, MT.

Bodoia, J.R. and J.F. Osterle. 1962, The development of free convection between heated vertical plates. Journal of heat transfer 11:40-43.

Incropera, F.P. and D.P Dewitt. 1981. Fundementals of heat transfer. John Wiley and sons, New York. 819 p.

Keane, R.E., S.F. Arno, and J.K. Brown. FIRESUM -- an ecological process model for fire succession in western conifer forests. USDA Forest Service Gen. Tech. Rep. INT-266. 76 p.

Luti, F.M. 1980. Transient flow development due to a strong heat source in the atmosphere part I: Uniform temperature source. Combustion Science and Tech. 23:163-175.

Luti, F.M. and T.A. Breustowski. 1977. Flow due to a two-dimensional heat source with cross flow in the atmosphere. Combustion Science and Tech. 16:71-87.

Quon, C. 1972. High rayleigh number convection in an enclosure - a numerical study. The physics of fluids 15(1):12-19.

Roache, P.J. 1972. Computational fluid dynamics. Hermosa Publishing, Albuquer que, New Mexico. 446 p.

Rothermel, R.C. and R.W. Mutch. 1986. Behavior of the life-threatening Butte fire: August 27-29, 1985. Fire Mgt. Notes 47(2):14-24.

Telesin, H.P. 1974. Flame radiation as a mechanism of fire spread in forests. In: Afgan, N.H. and J.M Beer (ed.) Heat transfer in flames, pp 441-449. John Wiley and Sons, New York, New York. 501 p.

Van Wagner, C.E. 1977. Conditions for the start and spread of crown fire. Can. J. For. Res. 7:23-34.

Williams, G.P. 1967. Thermal convection in a rotating fluid annulus part I: the basic axisymmetric flow. Journal Atmospheric Sci. 24:144-161

Appendix A

List of source code for the subroutine CROWN to be placed in the fire succession model FIRESUM. This subroutine computes crown fire ignition and tree mortality.

```
      SUBROUTINE crown(ntrees,dbh,kyr,ros,byram,flame,fzone,
     &                 ncrown,icwf)
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Th s subroutine determines the initiation of crown scorch in the         :
  ation stand.  A simulation grid monitiors wind speed and temp.        :
   e each tree is passed over the fire from a distance of 20 meters     :
to directly over the fire.                                               :
Definition and Units of some variables:                                  :
  Grid variables:                                                        :
    ux(i,j) - wind velocity x direction (m/sec)                          :
    vy(i,j) - wind velocity y direction (m/sec)                          :
    chi(i,j) - stream function vorticity (m2/sec)                        :
    e(i,j) - temperature (theta) dimensionless                           :
    w(i,j) - vorticity transport (1/sec)                                 :
  Equation constants and parameters:                                     :
    qi(i) - heat of ignition                                             :
    qf(i) - heat generated by fire at ignition points                    :
    gsize - gridsize (m)                                                 :
    iter - max number iterations                                         :
    conv - convergence limit or threshold                                :
    re - Renyolds number (stand. to 1)                                   :
    ge - Grashof number (stand to 1)                                     :
    pr - Prandtl's number                                                :
    dh - flame height increment (m)                                      :
    ix - max number grids in x direction (horiz.) starting in low left:
    iy - max number grids in y direction (vert.) starting in low left :
    chk(i,j) - iteration check array for all equations                   :
    ic(i) - x coordinate for check array                                 :
    jc(i) - y coordinate for check array                                 :
    dt - time interval (sec)                                             :
    fd - number of grid points in fire (flame) zone                      :
    wind - wind speed (km/hr)                                            :
    esf - energy smoothing function                                      :
    wsf - vorticity smoothing function                                   :
    ros - rate of spread of fire (m/min)                                 :
    flame - flame height (m)                                             :
    fwid - width of fire (m)                                             :
    fzone - flaming depth or zone in meters                              :
    delta - characteristic length for grid (dimensionless)               :
    byram - fire intensity (kw/m)                                        :
    t - temperature in deg C                                             :
    ftmp - flame temperature in deg C                                    :
    dbh - tree diameter (cm)                                             :
    kcons,pcons - Renyolds num. constants to compute Nusselt num.        :
    sbc - Stefan-Boltzmann constant (5.87x10-8)                          :
    ctok - centigrade to kelvin conversion factor                        :
    tg(i) - temperature at live and dead crown gridpoints                :
    f(i) - view factor                                                   :
    trc - temperature term for radiation and convection                  :
    tcond - temperature term for conduction computation                  :
    tcrw - temperature of ignition point (deg k)                         :
    carea - crown surface area receiving radiation - m2                  :
    ac - ignition point surface area (m2)                                :
    ab - ignition point x-section area (m2)                              :
    pie - pi (3.14159...)                                                :
    ipt - correct array index for air properties at right temp           :
    term1,term2 - terms in view factor equations                         :
    jjjj(i),iii - grid point array coordiates                            :
    dum1(i,j) - dummy array to calculate convergence                     :
    cbd(i) - live crown bulk density (kg/m3)                             :
```

```
c      vfl(i) - dead crown bulk density (kg/m3)                            :
c      cfmc(i) - live crown moisture content (prop)                        :
c      vfmc(i) - dead crown moisture content (prop)                        :
c      cflm(i) - live crown flammability factor                            :
c      csvr(i) - live crown surface area to volume ratio (1/m)             :
c      vsvr(i) - dead crown surface area to volume ratio (1/m)             :
c      cpd(i) - specific heat of wood (cal/g/deg c)                        :
c      tig - temperature of ignition                                       :
c      tboil - temperature of boiling point - degrees C                    :
c      epslon - fudge factor for smoothing functions                       :
c      ad(i) - air density (kg/m3)                                         :
c      sha(i) - specific heat of air (j/kg-deg K)                          :
c      kv(i) - kinematic viscosity (m2/sec)                                :
c      tc(i) - thermal conductivity (W/m-deg K)                            :
c      chtc(i) - convective heat transfer coefficient (W/m2-deg K)         :
c      bl(i) - typical needle length for species i (m)                     :
c      tcc - thermal conductivity of crown (W/m-degK)                      :
c      ncrown(i) - number trees ignited by height class                    :
c   Functions and subroutines called:                                      :
c      itable - computes air properties at various temps                   :
c      fltemp - computes the flame temperature                             :
c                                                                          :
c :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      common/oper/ ns,nspan,nruns,circut,nwrstr,ifire,sburn,ibr,impb
      common/leaf/aside(7),c(7),alpha(7),b2(7),b3(7),cext(8),crat(7),
     &            sigma(7),ap(7),betap(7)
      common/plotq/elev,rock,till,soilm,text,excess,pltsiz,ifg
      common/site/ occur(500),rh,wind,ttheta,t
      common/cfire/cbd(7),vfl(7),cfmc(7),vfmc(7),cflm(7),csvr(7),
     &             vsvr(7),bl(7)
      real dbh(3000),ntrees(3000)
      real ux(20,20),vy(20,20),chi(20,20),w(20,20),e(20,20),
     &     qi(2),duml(20,20),chk(5,5),ad(20),sha(20),kv(20),tc(20),
     &     qf(2),f(2),cpd(2),tcc(2),db(2)
      integer ic(5),jc(5),jjjj(2),ncrown(5)
      data wind/5.0/,ttheta/10.0/,t/22.0/,dx/1.0/,dy/1.0/
      data ros/3.0/,byram/200.0/,flame/1.3/,fzone/2.0/
      data ad /1.3947,1.1614,0.9950,0.8711,0.7741,0.6964,0.6329,
     &         0.5804,0.5356,0.4975,0.4643,0.4345,0.4097,0.3868,
     &         0.3666,0.3482,0.3166,0.2902,0.2679,0.2488/,
     &     sha/1006.0,1007.0,1009.0,1014.0,1021.0,1030.0,1040.0,
     &         1051.0,1063.0,1075.0,1087.0,1099.0,1110.0,1121.0,
     &         1131.0,1141.0,1159.0,1175.0,1189.0,1207.0/,
     &     kv /11.44E-6,15.89E-6,20.92E-6,26.41E-6,32.39E-6,38.39E-6,
     &         45.57E-6,52.69E-6,60.21E-6,68.10E-6,76.37E-6,84.93E-6,
     &         93.80E-6,102.9E-6,112.2E-6,121.9E-6,141.8E-6,162.9E-6,
     &         185.1E-6,213.0E-6/,
     &     tc /0.02230,0.02630,0.03000,0.03380,0.03730,0.04070,0.04390,
     &         0.04690,0.04970,0.05240,0.05490,0.05730,0.05960,0.06200,
     &         0.06430,0.06670,0.07150,0.07630,0.08200,0.09100/
      data tig/320.0/,tboil/100.0/,sbc/5.87E-8/,xlow/1E-25/,
     &     ctok/273.15/
      data gsize/1.0/,ix/20/,iy/20/,iter/1000/,conv/0.00001/,
     &     epslon/0.01/
      data cpd/0.4,0.266/,tig/320.0/,tboil/100.0/,sbc/5.87E-8/,
     &     cpdl/0.00116/,tcc/3.347E-6,1.25E-5/,ctok/273.15/,
     &     db/0.5,0.64/,pie/3.14159/

c ..... Initialize all variables
      umax = 0.0
```

```
      vmax = 0.0

C ..... Assign coordinates for iteration check array
      do 1 i = 1,5
          if(i .eq. 1) then
               ic(i) = 5
               jc(i) = iy/2
          elseif(i .eq. 2) then
               ic(i) = ix/2
               jc(i) = iy - 5
          elseif(i .eq. 3) then
               ic(i) = ix - 5
               jc(i) = iy/2
          elseif(i .eq. 4) then
               ic(i) = ix/2
               jc(i) = 5
          elseif(i .eq. 5) then
               ic(i) = ix/2
               jc(i) = iy/2
          endif
    1 continue
      do 4 j = 1,iy
           do 3 i = 1,ix
                e(i,j) = 0.0
    3      continue
    4 continue

C ..... Initialize wind-temperature simulation grid

      do 15 j = 1,iy
           do 10 i = 1,ix
                vy(i,j) = 0.0
                ux(i,j) = 1.0
                chi(i,j) = float(j)
                w(i,j) = 0.0
                if(e(i,j) .lt. 0.9 .or. e(i,j) .gt. 1.0) e(i,j) = 0.0
                if(j .eq. 1) then
                     ux(i,j) = 0.0
                     if(i .eq. 1) then
                          w(i,j) = 0.0
                     else
                          w(i,j) = 1.0
                     endif

C ..... Assign fire grid points based on flaming zone depth and length
                     if(i .eq. ix/2) then
                          e(i,j) = 1.0
                          fd = fzone/gsize
                          iii = ifix(fd + 0.5) + 1
                          jjj = 1
                          if(flame .gt. 0.5 .and. flame .le. 1.5)
     &                         jjj = 2
                          if(flame .gt. 1.5 .and. flame .le. 2.5)
     &                         jjj = 3
                          if(flame .gt. 2.5 .and. flame .le. 3.5)
                               jjj = 4
                          if(flame .gt. 3.5 .and. flame .le. 4.5)
     &                         jjj = 5
                          if(flame .gt. 4.5 .and. flame .le. 5.5)
     &                         jjj = 6
```

```
                      istrt = iii / 2
                      iengy = ix/2 - istrt
                      iend = iengy + iii - 1
                      iup = jjj
                      do 7 ii = 1,iii
                           do 6 jj = 1,jjj
                                ipos = (ix/2) - istrt + (ii - 1)
                                yred = 0.9+((1.0/float(jj))/10.0)
                                if(ipos .eq. ix/2) then
                                     e(ipos,jj) = 1.0 * yred
                                elseif(ipos .eq. ix/2+1 .or.
     &                          ipos .eq. ix/2-1) then
                                     e(ipos,jj) = 0.99 * yred
                                elseif(ipos .eq. ix/2+2 .or.
     &                          ipos .eq. ix/2-2) then
                                     e(ipos,jj) = 0.98 * yred
                                endif
    6                      continue
    7                 continue
                 endif
            elseif(j .eq. 2) then
                 if(i .eq. 1) then
                      w(i,j) = 0.0
                 else
                      w(i,j) = 0.0
                 endif
            endif
            dum1(i,j) = chi(i,j)
   10    continue
   15 continue

C ..... Calculate flame temperature
      ftmp = 1400.0

C ..... Calculate time interval variables
      umax = 1.0
      vmax = 0.0
      dt = 0.25

C ..... Calculate a characteristic length using gridsize
      delta = gsize

C ..... Start iteration for the simulation grid for all equations
      do 100 iii = 1,iter

C ..... Update the iteration check array
           do 18 j = 1,5
                do 17 i = 1,5
                     if(j .eq. 1) chk(i,j) = w(ic(i),jc(i))
                     if(j .eq. 2) chk(i,j) = e(ic(i),jc(i))
                     if(j .eq. 3) chk(i,j) = chi(ic(i),jc(i))
                     if(j .eq. 4) chk(i,j) = ux(ic(i),jc(i))
                     if(j .eq. 5) chk(i,j) = vy(ic(i),jc(i))
   17           continue
   18      continue

C ..... Calculate delta t or time interval of iteration
           if(iii .gt. 1) then
                dt = 1.0 / (umax / dx + vmax / dy)
           endif
```

```
C ..... Calculate steady state temp-velocity simulation grid
           do 30 j = 2,iy-1
                do 20 i = 2,ix-1

C ..... Calculate Renyolds, Prandtl and Grashof number for each grid
                     t1 = (t + (e(i+1,j) * (ftmp - t))) + ctok
                     t2 = (t + (e(i-1,j) * (ftmp - t))) + ctok
                     t0 = abs(max(t1,t2))
                     if(i .ge. iengy .and. i .le. iend .and.
     &               j .le. iup) then
                          t0 = (t + (e(iengy,j)*(ftmp-t)))+ctok
                     endif
                     ta = (t + (e(i,j) * (ftmp - t))) + ctok
                     ipt = itable(ta)
                     ipr = itable(t0)
                     re = (wind * sqrt(ux(i,j)**2.0 + vy(i,j)**2.0) *
     &                    delta) / (kv(ipt))
                     if(re .eq. 0.0) re = 1.0
                     if(re .gt. 500000.0) re = 500000.0
                     re = re * 1.0
                     ge = (9.81 * (ta - (t+ctok)) * delta**3.0) /
     &                    (t0 * kv(ipt)**2.0)
                     ge = abs(ge)
                     pr = (ad(ipr)*kv(ipr)*sha(ipr))/tc(ipr)

C ..... Calculate vorticity transport (wgrid) at interior points
                     w1 = (dt/2.0) * ((ux(i,j)*(w(i+1,j) - w(i-1,j))
     &                    / dx) + (vy(i,j)*(w(i,j+1) - w(i,j-1))
     &                    /dy))
                     w2 = (dt/re) * (((w(i+1,j) - (2.0*w(i,j)) +
     &                    w(i-1,j)) / dx**2.0) + ((w(i,j+1) -
     &                    (2.0*w(i,j)) + w(i,j-1)) / dy**2.0))
                     w3 = (dt*ge)/(2.0*re**2.0) * ((e(i+1,j) -
     &                    e(i-1,j)) / dx)
                     if(i .ge. 3 .and. i .le. 18 .and. j .ge. 3 .and.
     &               j .le. 18) then
                          wsf = epslon * (w(i+2,j) - 4.0*w(i+1,j) +
     &                          6.0*w(i,j) - 4.0*w(i-1,j) + w(i-2,j))
                     else
                          wsf = 0.0
                     endif
                     w(i,j) = w(i,j) - w1 + w2 + w3 + wsf
                     if(w(i,j) .le. xlow) w(i,j) = 0.0

C ..... Calculate energy (e) or temp at interior points
                     if(i .ge. iengy .and. i .le. iend .and.
     &               j .le. iup) then
                          e(i,j) = e(i,j)
                     else
                          e1 = (dt/2.0) * (((ux(i,j)*
     &                         (e(i+1,j) - e(i-1,j))) / dx) +
     &                         ((vy(i,j)*(e(i,j+1) - e(i,j-1)))
     &                         / dy))
                          e2 = (dt/(re*pr)) * (((e(i+1,j)-(2.0*e(i,j))
     &                         + e(i-1,j)) / dx**2.0) + ((e(i,j+1) -
     &                         (2.0*e(i,j)) + e(i,j-1)) / dy**2.0))
                          if(j .ge. 3 .and. j .le. 18 .and. i .ge.
     &                    3 .and. i .le. 18) then
                               esf = epslon * ((e(i+2,j) - 4.0*e(i+1,j)
```

```fortran
     &                         +  6.0*e(i,j) - 4.0*e(i-1,j) + e(i-2,j))
     &                         + (e(i,j+2) - 4.0*e(i,j+1) + 6.0*e(i,j)
     &                         - 4.0*e(i,j-1) + e(i,j-2)))
                    else
                         esf = 0.0
                    endif
                    e(i,j) = e(i,j) - e1 + e2 + esf
                    if(e(i,j) .le. xlow) e(i,j) = 0.0
               endif
   20     continue
   30 continue

C ..... Start iteration for the stream function equation
      do 80 ii = 1,iter

C ..... Find the stream function chi by iterative techniques
C ..... Estimate stream function for interior points
          do 50 j = 2,iy-1
               do 40 i = 2,ix-1
                    chi(i,j) = ((dx**2.0 * dy**2.0) / (2.0 *
     &                         (dx**2.0 + dy**2.0))) *
     &                         (((chi(i+1,j) + chi(i-1,j)) /
     &                         dx**2.0) + ((chi(i,j+1) +
     &                         chi(i,j-1)) / dy**2.0) -
     &                         w(i,j))
                    if(chi(i,j) .lt. xlow) chi(i,j) = 0.0
   40          continue
   50     continue

C ..... Find stream function bottem and top boundry values
          do 60 i = 1,ix
               chi(i,1) = 1.0
               chi(i,iy) = chi(i,iy-1) + 1.0
   60     continue

C ..... Find stream function side boundry values
          do 65 j = 1,iy
               chi(1,j) = float(j)
               chi(ix,j) = chi(ix-1,j)
   65     continue

C ..... Test to see if the matrix has converged
          iflag = 0
          do 75 j = 1,iy
               do 70 i = 1,ix
                    diff = abs(chi(i,j)-dum1(i,j))
                    if(diff .gt. conv) then
                         iflag = 1
                    endif
                    dum1(i,j) = chi(i,j)
   70          continue
   75     continue
          if(iflag .eq. 0) go to 81
   80 continue
      write(6,1000) iii

C ..... Compute new wind velocity vectors (ux,vy) from stream function
C ..... for the interior grid points
   81 do 83 j = 2,iy-1
          do 82 i = 2,ix-1
```

```
                    ux(i,j) = (chi(i,j+1) - chi(i,j-1))/(2.0*dy)
                    vy(i,j) =-(chi(i+1,j) - chi(i-1,j))/(2.0*dx)
                    if(ux(i,j) .le. xlow) ux(i,j) = 0.0
                    if(vy(i,j) .le. xlow) vy(i,j) = 0.0
                    if(ux(i,j) .gt. umax) umax = ux(i,j)
                    if(vy(i,j) .gt. vmax) vmax = vy(i,j)
   82           continue
   83       continue

C ..... Compute new wind velocities for top and bottom boundry points
            do 84 i = 1,ix
                 ux(i,1) = (chi(i,2) - chi(i,1)) / delta
                 ux(i,iy) = (chi(i,iy) - chi(i,iy-1)) / delta
                 if(i .ge. 2 .and. i .le. ix-1) then
                      vy(i,1) = (chi(i+1,1)-chi(i-1,1)) / (2.0*delta)
                      vy(i,iy) = (chi(i+1,iy)-chi(i-1,iy))/(2.0*delta)
                      if(ux(i,j) .gt. umax) umax = ux(i,j)
                      if(vy(i,j) .gt. vmax) vmax = vy(i,j)
                 endif
   84       continue

C ..... Compute new wind velocities for side boundry points
            do 85 j = 1,iy
                 vy(1,j) = -(chi(2,j) - chi(1,j)) / delta
                 vy(ix,j) = -(chi(ix,j) - chi(ix-1,j)) / delta
                 if(j .ge. 2 .and. j .le. iy-1) then
                      ux(1,j) = (chi(1,j+1)-chi(1,j-1)) / (2.0*delta)
                      ux(ix,j) = (chi(ix,j+1)-chi(ix,j-1))/(2.0*delta)
                      if(ux(i,j) .gt. umax) umax = ux(i,j)
                      if(vy(i,j) .gt. vmax) vmax = vy(i,j)
                endif
            continue

C ..... Calculate new boundary values for vort and energy
            do 86 i = 1,ix

C ..... Calculate vorticity transport (w) at top and bottom
                 w(i,1) = 2.0 * ((chi(i,2) - chi(i,1)) / delta**2.0)
                 w(i,iy) = 0.0

C ..... Calculate energy (e) or temp at top and bottem points
                 if(i .lt. iengy .and. i .gt. iend) then
                      e(i,1) = e(i,1)
                 endif
                 e(i,iy) = e(i,iy-1)
   86       continue

C ..... Calculate new boundary values for vort and energy for sides
            do 90 j = 1,iy
                 w(1,j) = 0.0
                 w(ix,j) = w(ix-1,j)

C ..... Calculate energy (e) or temp at interior points
                 e(1,j) = 0.0
                 e(ix,j) = e(ix-1,j)
            continue

C ..... Compare iterative check array for equilibrium
            iflag = 0
            do 98 j = 1,5
```

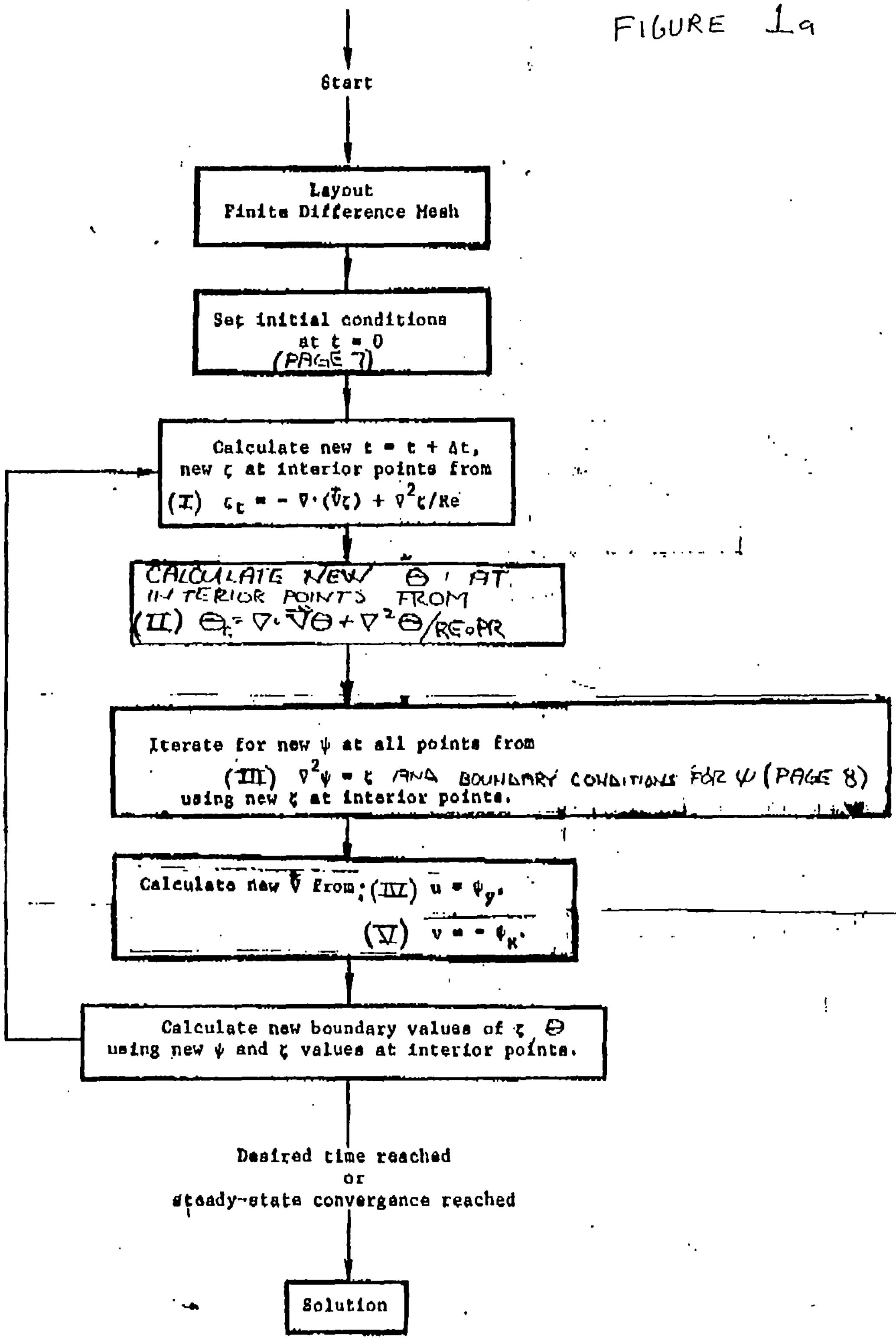
```
            do 97 i = 1,5
                 if(j .eq. 1) diff = abs(w(ic(i),jc(i))-chk(i,j))
                 if(j .eq. 2) diff = abs(e(ic(i),jc(i))-chk(i,j))
                 if(j .eq. 3) diff = abs(chi(ic(i),jc(i))-chk(i,j))
                 if(j .eq. 4) diff = abs(ux(ic(i),jc(i))-chk(i,j))
                 if(j .eq. 5) diff = abs(vy(ic(i),jc(i))-chk(i,j))
                 if(diff .gt. conv) then
                      iflag = iflag + 1
                 endif
   97       continue
   98  continue
       if(iflag .le. 3) go to 101
  100 continue

C ..............................................................
C ..... Start the Crown fire initiation process for each species
C ..............................................................
  101 do 160 i= 1,ns

C ..... Compute the heat of ignition for live and dead crown
       qi(1) = ((cpd(1) + (cpdl*(tig - t)/2)) * (tig - t)) +
     &         (cfmc(i) * ((tboil - t) + 540.0))
       qi(2) = ((cpd(2) + (cpdl*(tig - t)/2)) * (tig - t)) +
     &         (vfmc(i) * ((tboil - t) + 540.0))
       qi(1) = exp(-138.0 / csvr(i)) * cbd(i) * qi(1)
       qi(2) = exp(-138.0 / vsvr(i)) * vfl(i) * qi(2)

C ..... Compute tree specific and simulation specific parameters
       ispp = i
       cratio = crat(ispp)
       ni = ntrees(ispp)
       jj = isum(ntrees,ispp-1)
       dt = gsize / ros

       do 150 j = 1,ni

C ..... Compute the height of live crown, dead crown and tree
            ikk = jj + j
            h = (137.0 + b2(i)*dbh(ikk) - b3(i)*dbh(ikk)**(2.0))
     &          / 100.0
            ch = h - (cratio * h)
            if(ch .lt. 0.0) ch = 0.0
            vfh = ch - (0.2 * cratio * h)
            if(vfh .lt. 0.0) vfh = 0.0
            fwid = pltsiz**0.5
            carea = 1.0

C ..... Move individual tree towards fire
            jjjj(1) = ifix(ch + 0.5)
            if(jjjj(1) .gt. iy) jjjj(1) = iy
            jjjj(2) = ifix(vfh + 0.5)
            if(jjjj(2) .gt. iy) jjjj(2) = iy
            fdist = gsize * float(ix/2)
            tcrw = t
            do 140 k = 1,ix/2+1

C ..... Calculate the distance from the fire to the tree
                 xdist = fdist - float(k-1)*gsize

C ..... Calculate the view factor
```

```
                f1 = ((xdist**2.0)*carea)/flame
                f2 = ((xdist**2.0)*carea)/flame
                dh = flame/10.0
                f(1) = 0.0
                f(2) = 0.0
                do 110 m = 1,10
                     term1 = (ch - (float(m)*dh))**2.0
                     term2 = (vfh - (float(m)*dh))**2.0
                     f(1) = f(1) + ((dh/(xdist**2.0+term1))*
     &                      ((1.0 / (xdist**2.0 + term1 +
     &                      (fwid/2.0)**2.0)) + ((1.0 /
     &                      (xdist**2.0 + term1)**0.5) *
     &                      (atan((fwid/2.0) / (xdist**2.0
     &                      + term1)**0.5)))))
                     f(2) = f(2) + ((dh/(xdist**2.0+term2))*
     &                      ((1.0 / (xdist**2.0 + term2 +
     &                      (fwid/2.0)**2.0)) + ((1.0 /
     &                      (xdist**2.0 + term2)**0.5) *
     &                      (atan((fwid/2.0) / (xdist**2.0
     &                      + term2)**0.5)))))
 110            continue
                f(1) = f(1) * f1
                f(2) = f(2) * f2

C ..... Calculate nearest coordinates to crown area in sim grid
                iii = (ix/2) + ifix(xdist+0.5)

C ..... Compute temperatures at this grid point
                do 120 n = 1,2
                     ac = pie * (db(n)/100.0) * bl(i)
                     ab = (pie * (db(n)/100.0)**2.0) / 4.0
                     tgrid = (t + (e(iii,jjjj(n)) * (ftmp - t)))
     &                       + ctok

C ..... Compute air properties and equation constants at grid point
                     ipt = itable(tgrid)
                     re = (wind * sqrt(ux(iii,jjjj(n))**2.0 +
     &                    vy(iii,jjjj(n))**2.0) *
     &                    (db(n) / 100.0)) / kv(ipt)
                     if(re .gt. 4.0 .and. re .le. 40.0) then
                          kcons = 0.911
                          pcons = 0.385
                     elseif(re.gt.40.0 .and. re.le.4000.0) then
                          kcons = 0.683
                          pcons = 0.466
                     elseif(re .gt. 4000.0 .and. re .le.
     &               40000.0) then
                          kcons = 0.193
                          pcons = 0.618
                     elseif(re .gt. 40000.0 .and. re .le.
     &               400000.0) then
                          kcons = 0.027
                          pcons = 0.805
                     else
                          print *,'re: ',re
                          stop
                     endif
                     pr = (ad(ipt)*kv(ipt)*sha(ipt))/tc(ipt)
                     chtc = (kcons*(re**pcons)*pr**(0.3333)) *
     &                      (tc(ipt) / (db(n)/100.0))
```

```
C ..... Compute temperature of crown through heat transfer
                         trc = ((dt/(cbd(i)*cpd(n))) *
     &                         (ftmp**4.0+(tgrid*chtc)/(sbc*f(n)) +
                               (t*tcc(n)*ab)/(sbc*ac*f(n)*
                               bl(i))))
                         tcond = ((dt/(cbd(i)*cpd(n))) *
     &                           (tcrw**4.0+tcrw-((h/(sbc*f(n))) +
     &                           (tcc(n)*ab)/
     &                           (sbc*ac*f(n)*bl(i)))))
                         tcrw = tcrw + trc - tcond
                         qf(n) = sbc*ac*f(n)*(ftmp**4.0 - tcrw**4.0) +
     &                           chtc*ac*(tgrid - tcrw) -
     &                           tcc(n)*ab*(tcrw - t)/bl(i)
  120               continue

C ..... Compare heat transfer to crown with heat of ignition
                    do 130 n = 1,2
                         if(qf(n) .gt. qi(n)) then
                              if(h .lt. 2.0) then
                                  ncrown(1) = ncrown(1) + 1
                              elseif(h.ge.2.0 .and. h.lt.5.0) then
                                  ncrown(2) = ncrown(2) + 1
                              elseif(h.ge.5.0 .and. h.lt.10.0) then
                                  ncrown(3) = ncrown(3) + 1
                              elseif(h.ge.10.0 .and. h.lt.15.0) then
                                  ncrown(4) = ncrown(4) + 1
                              elseif(h.ge.15.0) then
                                  ncrown(5) = ncrown(5) + 1
                              endif
                              go to 150
                         endif
  130               continue
  140          continue
  150     continue
  160 continue
      stop
 1000 format(1h ,'the stream function did not converge. Iteration ',
     &       i5)
      end


      FUNCTION itable(t)
C :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C This function computes the various properties of air at a          :
C specified temperature level in degrees kelvin.                      :
C :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

      if(t .ge. 0.0 .and. t .lt. 250.0)     itable =  1
      if(t .ge. 250.0 .and. t .lt. 300.0)   itable =  2
      if(t .ge. 300.0 .and. t .lt. 350.0)   itable =  3
      if(t .ge. 350.0 .and. t .lt. 400.0)   itable =  4
      if(t .ge. 400.0 .and. t .lt. 450.0)   itable =  5
      if(t .ge. 450.0 .and. t .lt. 500.0)   itable =  6
      if(t .ge. 500.0 .and. t .lt. 550.0)   itable =  7
      if(t .ge. 550.0 .and. t .lt. 600.0)   itable =  8
      if(t .ge. 600.0 .and. t .lt. 650.0)   itable =  9
      if(t .ge. 650.0 .and. t .lt. 700.0)   itable = 10
      if(t .ge. 700.0 .and. t .lt. 750.0)   itable = 11
      if(t .ge. 750.0 .and. t .lt. 800.0)   itable = 12
```

```
if(t .ge. 800.0 .and. t .lt. 850.0)   itable = 13
if(t .ge. 850.0 .and. t .lt. 900.0)   itable = 14
if(t .ge. 900.0 .and. t .lt. 950.0)   itable = 15
if(t .ge. 950.0 .and. t .lt. 1000.0)  itable = 16
if(t .ge. 1000.0 .and. t .lt. 1100.0) itable = 17
if(t .ge. 1100.0 .and. t .lt. 1200.0) itable = 18
if(t .ge. 1200.0 .and. t .lt. 1300.0) itable = 19
if(t .ge. 1300.0 .and. t .lt. 1400.0) itable = 20
if(t .ge. 1400.0)                     itable = 20
return
END
```

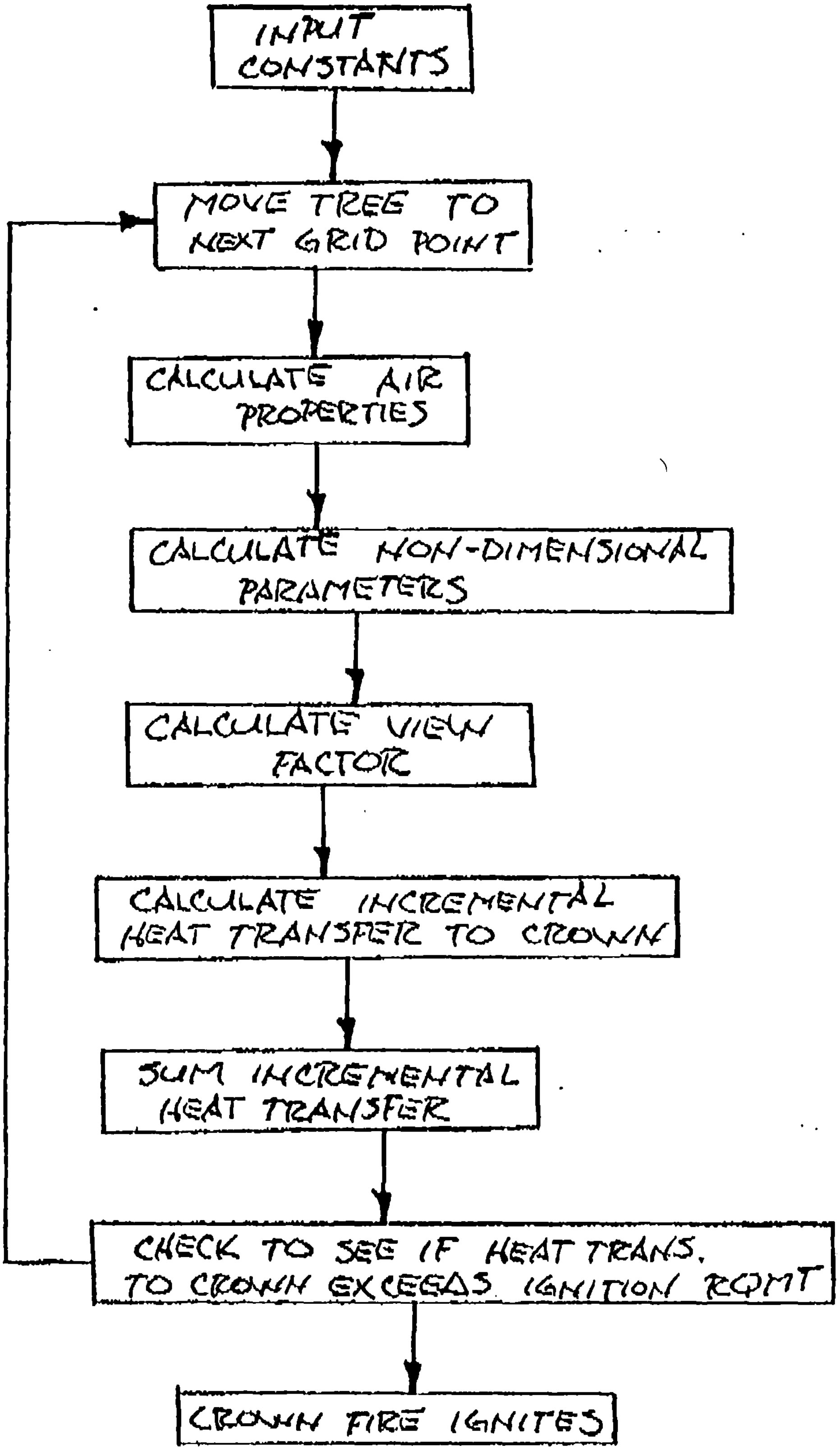
FIGURE 1a



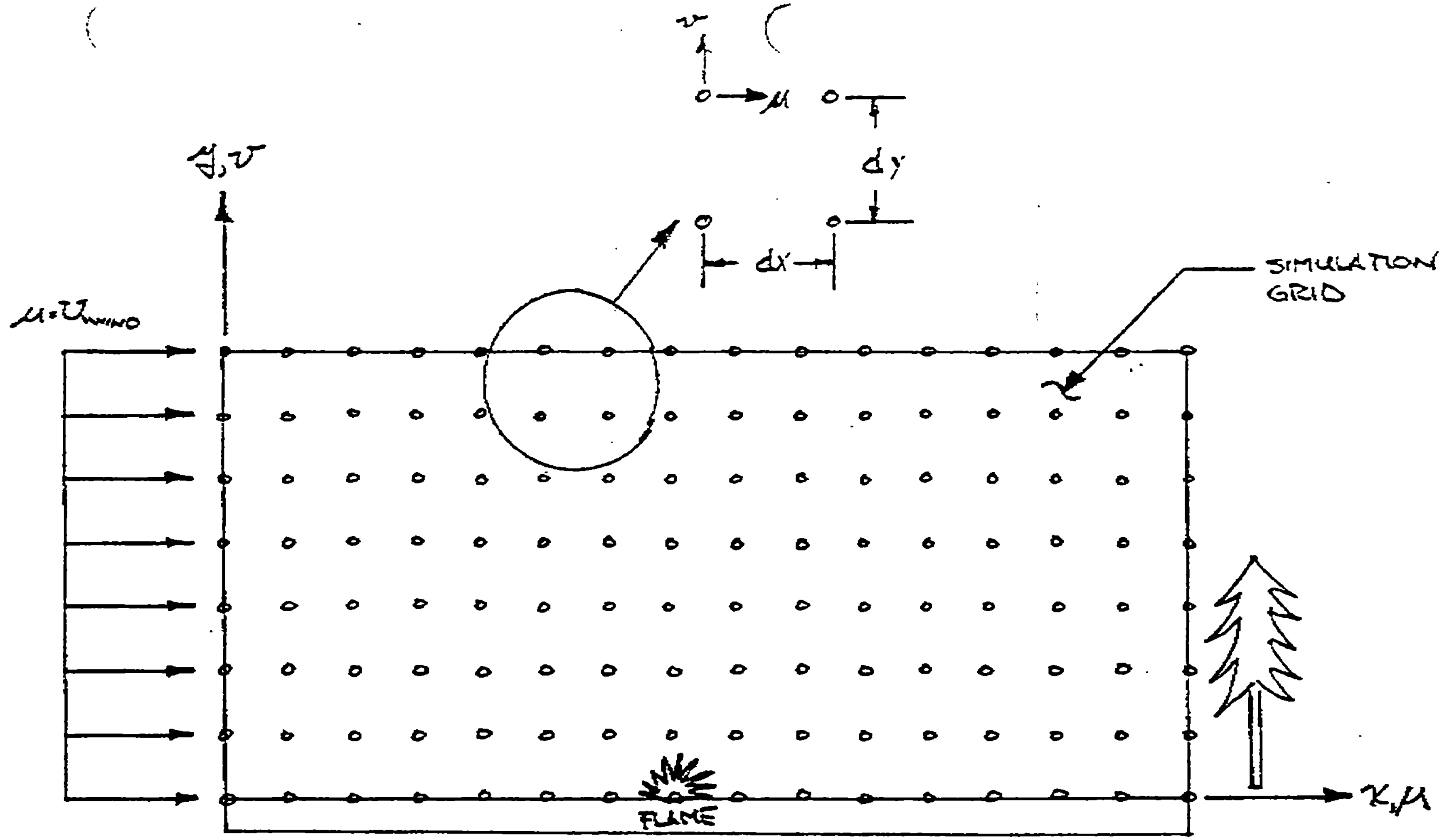
TEMPERATURE - VELOCITY
CALCULATION ALGORITHM

FIGURE 1b

INITIALLY THE 1ST TREE IS OUTSIDE THE GRID



RADIATION - CONVECTION CALCULATION ALGORITHM



FINITE DIFFERENCE TEMPERATURE VELOCITY GRID.

FIGURE 2

FIGURE 3

1.) VORTICITY TRANSPORT

$$\zeta_{i,j}^{(n+1)} = \zeta_{i,j}^{(n)} - \Delta t \left[u_{i,j}^{(n)} \left(\frac{\zeta_{i+1,j}^{(n)} - \zeta_{i-1,j}^{(n)}}{2\Delta x} \right) + v_{i,j}^{(n)} \left(\frac{\zeta_{i,j+1}^{(n)} - \zeta_{i,j-1}^{(n)}}{2\Delta y} \right) \right]$$

$$+ \frac{\Delta t}{RE} \left[\frac{\zeta_{i+1,j}^{(n)} - 2\zeta_{i,j}^{(n)} + \zeta_{i-1,j}^{(n)}}{\Delta x^2} + \frac{\zeta_{i,j+1}^{(n)} - 2\zeta_{i,j}^{(n)} + \zeta_{i,j-1}^{(n)}}{\Delta y^2} \right] + \Delta t \frac{GR}{(RE)^2} \left[\frac{\Theta_{i+1,j}^{(n)} - \Theta_{i-1,j}^{(n)}}{2\Delta x} \right]$$

2.) ENERGY

$$\Theta_{i,j}^{(n+1)} = \Theta_{i,j}^{(n)} - \Delta t \left[u_{i,j}^{(n)} \left(\frac{\Theta_{i+1,j}^{(n)} - \Theta_{i-1,j}^{(n)}}{2\Delta x} \right) + v_{i,j}^{(n)} \left(\frac{\Theta_{i,j+1}^{(n)} - \Theta_{i,j-1}^{(n)}}{2\Delta y} \right) \right]$$

$$+ \frac{\Delta t}{RE \cdot PR} \left[\frac{\Theta_{i+1,j}^{(n)} - 2\Theta_{i,j}^{(n)} + \Theta_{i-1,j}^{(n)}}{\Delta x^2} + \frac{\Theta_{i,j+1}^{(n)} - 2\Theta_{i,j}^{(n)} + \Theta_{i,j-1}^{(n)}}{\Delta y^2} \right]$$

3.) STREAM FUNCTION - VORTICITY

$$\psi_{i,j} = \frac{1}{4} \left[\left(\frac{\psi_{i+1,j}^{(n)} - 2\psi_{i,j}^{(n)} + \psi_{i-1,j}^{(n)}}{\Delta x^2} \right) + \left(\frac{\psi_{i,j+1}^{(n)} - 2\psi_{i,j}^{(n)} + \psi_{i,j-1}^{(n)}}{\Delta y^2} \right) + \zeta_{i,j}^{(n)} \right]$$

STREAM FUNCTION

4.) $$u_{i,j}^{(n)} = \frac{\psi_{i,j+1}^{(n)} - \psi_{i,j-1}^{(n)}}{2\Delta y}$$

5.) $$v_{i,j}^{(n)} = \frac{\psi_{i+1,j}^{(n)} - \psi_{i-1,j}^{(n)}}{2\Delta x}$$

FINITE DIFFERENCED NAVIER-STOKES AND ENERGY EQNS (NON-DIMENSIONAL)

GREEK SYMBOLS

ζ - VORTICITY
Θ - TEMPERATURE
ψ - STREAM FUNCTION

u - X VELOCITY
v - Y VELOCITY
t - time
x - X DISTANCE
y - Y DISTANCE

RE - REYNOLD'S NO.
GR - GRASHOFF NO.
PR - PRANDTL NO.

FIGURE 4

ALONG THE BASE:

1.) $\psi'_{i,1} = 1$

2.) $\zeta'_{i,1} = 2\left[\frac{\psi'_{i,2} - \psi'_{i,1}}{(\Delta y)^2}\right]$

3.) $\Theta_{i,1} = 0$ OR $\Theta_{i_F,1} = 1$

ALONG THE LEFT HAND SIDE:

1.) $\psi'_{1,j} = j$

2.) $\zeta' = 0$

3.) $\Theta_{1,j} = 0$

ALONG THE RIGHT HAND SIDE

1.) $\psi'_{L,j} = \psi'_{L-1,j}$

2.) $\zeta'_{L,j} = \zeta'_{L-1,j}$

3.) $\Theta_{L,j} = \Theta_{L-1,j}$

ALONG UPPER SURFACE

1) $\psi'_{i,M} - \psi'_{i,M-1} = 1$

2) $\zeta' = 0$

3.) $\Theta_{i,M} = \Theta_{i,M-1}$

TEMPERATURE - VELOCITY
GRID BOUNDARY CONDITIONS

$$\frac{Q_c^{n+1}}{A_c} = \Delta t \left[\sigma F^{n+1} \left[T_F^4 - (T_c^n)^4 \right] + h^{n+1} \left(T_{GRID}^{n+1} - T_c^n \right) - \frac{k_B D_B}{4 L_B^2} \left(T_c^n - T_0 \right) \right]$$

RADIATION - CONVECTION HEAT TRANSFER EQUATION

$\left(\frac{Q_c}{A_c}\right)^{n+1}$ IS THE AMOUNT OF HEAT TRANSFERED TO THE CROWN DURING TIME INTERVAL Δt AS THE TREE MOVES FROM ONE GRID POINT TO THE NEXT. WE MUST SUM THE HEAT TRANSFER FROM EACH GRID POINT AS THE TREE MOVES ALONG.

L_B - AVERAGE BRANCH LENGTH (M)

D_B - AVERAGE BRANCH DIAMETER (M)

σ - STEFAN-BOLTZMANN CONSTANT $\sigma = 5.670 \times 10^{-8}$ (WATT/M^2-°K^4)

k_B - THERMAL CONDUCTIVITY OF BRANCH - (WATT/M-°K)

$C_{p,c}$ - SPECIFIC HEAT OF CROWN MATERIAL - (JOULE/KG-°K)

m_c - MASS OF CROWN MATERIAL - (KG)

T_c - TEMP. OF CROWN MATERIAL - (°K)

T_0 - AMBIENT TEMP OF AIR (CONSTANT) - (°K)

T_F - FIRE TEMP. (CONSTANT) - (°K)

T_{GRID} - LOCAL GRID TEMP. ADJACENT - (°K)
TO CROWN WHERE $T_{GRID} = \Theta(T_F - T_0) + T_0$

h - CONVECTIVE HEAT TRANSFER COEFFICIENT - (WATT/M^2-°K)
(FROM TEMP/VEL. GRID TO CROWN)

F - VIEW FACTOR

FIGURE 6

Crown characteristics in model

Live and dead crown

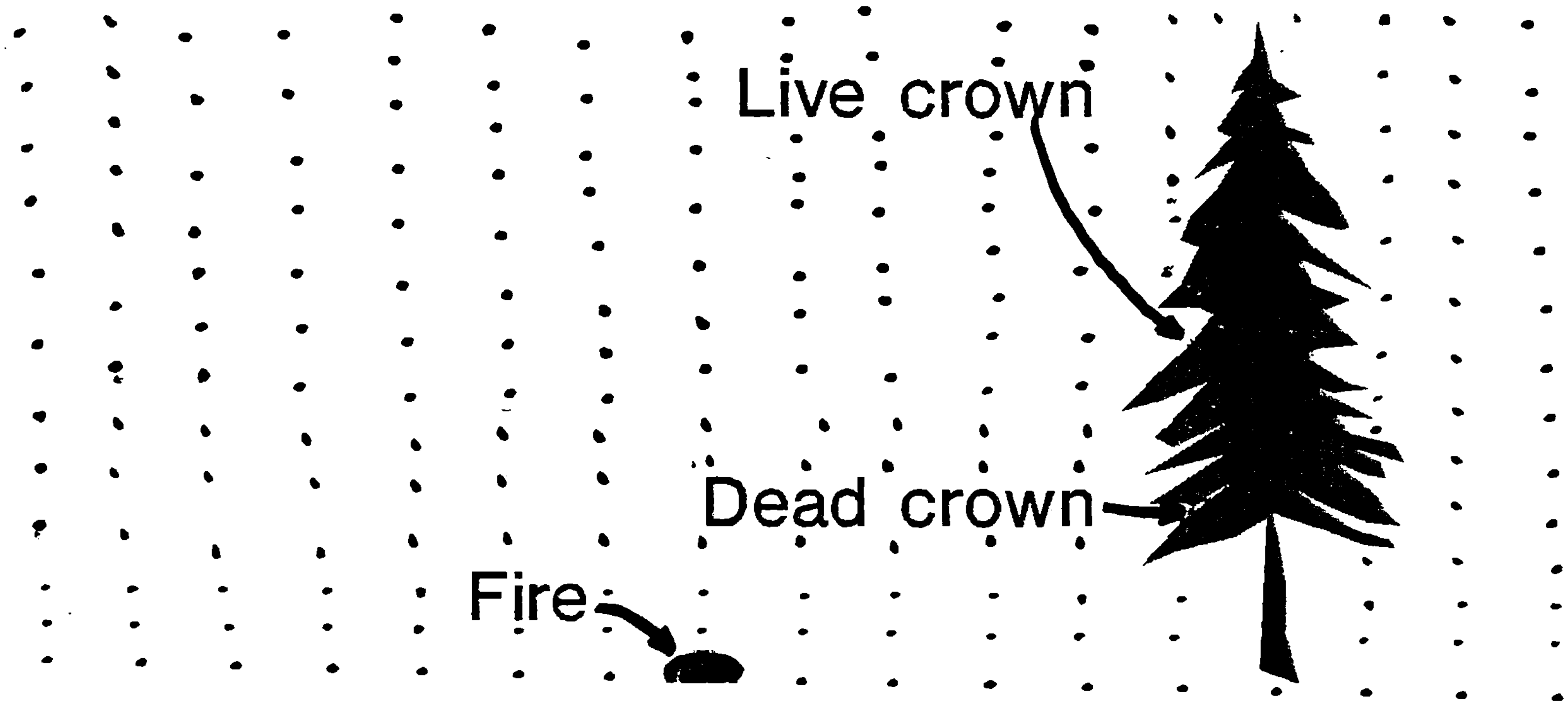


FIGURE 7

$$Q_i = \left\{ \left[C_{pw} + \left(0.00116 \left(T_i - T_c \right) / 2.0 \right) \right] \left(T_i - T_c \right) + \left[CM_c \left[\left(T_b - T_c \right) + 540.0 \right] \right] \right\} e^{\left[-138 / SVR \right]} \rho_c$$

where

Q_i - Heat needed to ignite crown (Joules)

C_{pw} - Specific heat of crown material (live or dead) (Joule/Kg-°K)

T_i - Temperature of ignition (°K)

T_c - Temperature of crown (°K)

CM_c - Crown moisture content (live or dead) (%)

SVR - Surface area to volume ratio (m^2/m^3) (live or dead crown)

ρ_c - Bulk density of crown (live or dead) (Kg/m^3)